中国散裂中子源上的一种双束团数据解析方法

甘林 [1]　刘小桐 [1]　蒋伟 [2,3]
1（深圳信息职业技术学院微电子学院　深圳 518172）
2（中国科学院高能物理研究所　北京 100049）
3（中国散裂中子源　东莞 523803）

摘要　中国散裂中子源（CSNS）可提供 0.3 eV 至 300 MeV 的白光中子束流，总束流强度可达 10^7n/s/cm^2，为中子俘获反应截面的测量提供了一个优秀的实验平台。CSNS 在正常的运行模式下会由加速器产生两个间隔 410 ns 的质子束团先后打靶，因此产生的中子束流也由间隔为 410 ns 的两个束团混合而成。为避免两个束团的效应相互干扰，影响中子俘获截面的能量精度，需要对实验数据进行解析和重构，还原单个束团的效应。现有的解析方法可以得到非常精细的解谱结果，但相对复杂，具有一定使用门槛。本工作提出了一种简化的双束团解谱方法，在保证中子能量精度的情况下适用于中子能量低于 1.2 MeV 的数据，为同类型的实验工作提供一种新的数据处理思路。

关键词　CSNS；双束团解析方法；中子俘获反应
中图分类号　TL99（建议原子能技术类的中图分类号）

A Simplified method for unfolding double-bunch data at CSNS

GAN Lin[1]　LIU Xiaotong[1]　JIANG Wei[2,3]
1(*Shenzhen Institute of Information Technology, Shenzhen 518172, China*)
2(*Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China*)
3(*Spallation Neutron Source Science Center, Dongguan 523803, China)*

Abstract　**[Background]**: The China Spallation Neutron Source (CSNS) provides a white neutron beam with an energy range from 0.5 eV to 300 MeV and a total beam intensity of up to 10^7n/s/cm^2, serving as an excellent experimental platform for the measurement of neutron capture reaction cross sections. During normal operation, the CSNS generates two proton bunches separated by 410 ns, consecutively striking the target, resulting in a mixed neutron beam composed of two bunches with a 410 ns interval. To avoid interference between the effects of the two bunches and maintain the energy precision of neutron capture cross sections, experimental data need to be analyzed and reconstructed to restore the effects of individual bunches. **[Purpose]**: The existing parsing method can yield very refined unfolding results, but it is relatively complex and has a certain usage threshold. Therefore, a more convenient data processing method needs to be found. **[Methods]**: This work utilized mathematical operations to analyze and reconstruct the data, with 410 ns as the unit time, and processed the data with a channel width of 4100 ns. Additionally, a comparison was made of the impacts of this method and existing methods on the accuracy of neutron incident energy. **[Results]**: This work proposes a simplified data processing method that achieves the same energy resolution as existing methods in the low-to-medium energy range, providing a new data processing approach for similar experimental work. **[Conclusions]**: The simplified data processing method presented in this study effectively

———————————————

基金名称项目(批准文号) 资助
第一作者：甘林，男，1988 年出生，2017 年于中国原子能科学研究院获博士学位，研究领域：粒子物理与原子核物理
通讯作者：刘小桐，E-mail: liuxt8841@outlook.com
收稿日期：20XX-00-00，修回日期：20XX-00-00

addresses the issue of excessive computational costs in analyzing low to medium energy neutron data from the CSNS. It offers a practical solution for experimental work requiring accurate analysis of neutron capture reactions in this energy range.

Key words　CSNS, Double-bunch unfolding method, Neutron capture reactions

中国散裂中子源（CSNS）位于广东省东莞市，是我国第一个白光中子源实验平台，为中国乃至全球的科研工作者提供了一个强大的研究平台[1-2]。CSNS 的中子束流是利用加速器加速并经过磁场偏转后的高能质子束流轰击散裂靶产生，其加速器频率为 25 Hz，质子能量约为 1.6 GeV。CSNS 装置概况如图 1 所示，与质子束流打靶方向呈 180°的角度引出一条中子束流，称之为反角白光中子束流线（back_n），包含了从 0.3 eV 至 300 MeV 范围内能量连续的中子，最大中子通量可达 10^7n/cm^2/s。距离打靶点 55 米和 76 米处分别设置有 ES1#1 和 ES#2 两个实验终端，其中 ES1#1 具有更高的中子束流强度，而 ES#2 具有更高的中子能量分辨。根据需求，用户可以选择合适的终端开展实验[3-6]。自 2018 年打靶出束以来，为基础物理、材料科学等多个学科的实验提供了束流支持[7-9]。

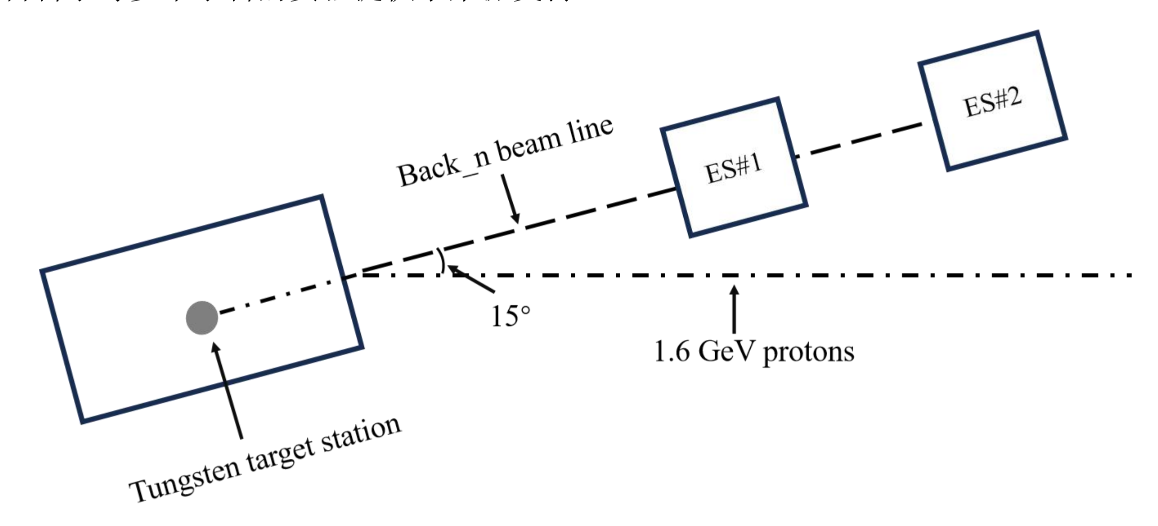


图 1　CSNS 装置示意图，引自文献[9]。
Fig.1　Schematic Diagram of the CSNS Facility, cited from Ref[9].

在常规运行时，加速器采用的是双束团模式，即两个相同的质子束团以 410 ns 为间隔轰击散裂靶。因此在实验中测量到的是两个间隔为 410 ns 的反应效应的叠加。实验中的中子入射能是根据飞行时间计算的，时间的起点为第一个质子束团的打靶时间，第二个中子束团中相同能量的中子对应的飞行时间则对应增加了 410ns。在测量中子俘获反应截面时，两个束团的叠加会导致计算出的中子的入射能精度下降，对于中高能的中子影响尤为显著。因此，为了得到准确的中子俘获反应激发函数，双束团的数据必须进行解析重构，还原出单个个中子束团真实的效应。

中科院高能物理研究所的易晗等人在 2020 年针对 CSNS 双束团数据开发了一套解析程序[10]，可以得到精细的中子反应截面，用户可以在此网址下载编译（http://code.ihep.ac.cn/yih/csns-back-n-doublebunchunfolder）。使用时需要先将测量得到的数据转化成直方图样式的 root 文件，再将该程序嵌入到数据处理程序中，经过 30 次至上百次的反复迭代，最终得到飞行时间单的一维谱或者飞行时间与能量的二维谱。为保证解谱后的数据能还原真实的反应产额，推荐直方图的时间道宽不超过延时的十分之一，即 41 ns。传统的中子俘获截面的测量使用 C_6D_6 液体闪烁探测器等具有较高伽马射线灵敏度的探测器，测量过程中获取反应中产生的所有伽马射线，数据处理时通过权重因子计算出中子俘获反应的比

例，进而得到目标反应的截面。因此探测器获取的统计量通常足够大，即使将时间谱的道宽取非常小的数值，也可保证较小的统计误差。这样的处理方法可以得到非常精细的激发函数曲线（中子俘获反应截面随入射能变化的曲线），但也会带来巨大的计算量，因此程序的作者也建议数据解析的时间跨度不超过 10^5 ns，否则可能会导致电脑内存错误，程序无法运行。此外，该程序的基于 Linux 系统开发，且使用方法相对复杂，具有一定的门槛，对不熟悉 Linux 系统以及 root 程序代码操作的用户有一定难度。

对于两种情形：1) 实验数据的统计量不足，使用较小的时间道宽意味着每道的统计量很小，会导致统计误差过大；2) 测量低能中子的数据，即飞行时间在 10^5 ns 至 10^6 ns 区间，宽道宽对能量误差的影响变得可以忽略，而继续使用较小的时间道宽意味着极大的运算量，不具有必要性。因此，上述两种情形更适合采用较大的时间道宽，此时选择简化的解谱方法可以在符合实验精度的前提下，实现数据解析的目的。本工作以 4100 ns 作为时间窗口为例，提出了一种双束团数据的简化解析方法，为同类型的中子俘获反应截面测量提供一种新的数据处理方案。

1 双束团数据解析原理

Back_n 的中子束流可以看做是两个能量分布相同，但是前后相差 410 ns 的中子束团叠加，分别称为束团 A 和束团 B。探测器的时间以束团 A 产生的时刻为起点。因此，在探测器时间 0 ~ 410 ns 内，探测器获取的数据全部来自于束团 A 在这个时间段的效应；在探测器时间 410 ~ 820 ns 内，则包括了束团 A 在第二个时间段的数据以及束团 B 在第一个时间段内的数据。依次类推，在之后的每个 410 ns 时间段内，均包括了束团 A 在该时间段内的数据以及束团 B 在上一个时间段内的数据。以 410 ns 为单位时间，假设单个束团在第 n 个单位时间段内产生的数据为 f(n)，探测器在对应时间段内获取的数据为 A(n)。那么有：

$$A(1) = f(1)$$
$$A(2) = f(1) + f(2)$$
$$\dots$$
$$A(n+1) = f(n) + f(n+1) \quad (1)$$

由于时间窗口为 4100 ns，每组数据包括了 10 个时间段。令其中任意一组数据的时间起点为第 x 个时间段的起点。单个束团在该时间窗口内的数据之和 $S=f(x)+f(x+1)+f(x+2)+\dots+f(x+9)$。由式 (1) 可知，$f(x)+f(x+1)=A(x+1)$，$f(x+2)+f(x+3)=A(x+3)$，…，$f(x+8)+f(x+9)=A(x+9)$。因此单个束团的数据与探测器获取的数据之间的关系式可以改写为：

$$S = A(x+1) + A(x+3) + A(x+5) + A(x+7) + A(x+9) \quad (2)$$

由式 (2) 可知，间隔取出数据再重新组合就可以得到该组数据中单个束团的数据。此外，由于探测器对于不同能量的伽马射线具有不同的探测效率，还需要取出每组数据的能量单谱。

本工作中将数据解析过程分为四个步骤：1) 将存储在 root 文件中的时间与能量二维谱导出为文本文档；2) 以 410 ns 为时间间隔将文档拆分成多个子文档；3) 对每组数据间隔取出子文档，合并成一个新的文档；4) 获取每组数据的能量单谱。

此外，由式 (1)和式 (2)可知，本方法最小的时间道宽为 820 ns。

2 双束团数据解析过程

2.1 导出 root 数据为文本文档

由于不同数据获取系统生成的 root 文件具有不同 Branch 结构，此处以本团队的数据结构为例。其中，能量和时间的 Branch 名称分别为 E 和 DeltaT，数据类型分别是 Double_t 和 ULong64_t。分别定义两个对应数据类型的变量 E1 和 DT1 用于取出 root 文件中的能量和时

间数据。完整的数据处理代码见附录 1。代码的思路是逐事件读取 root 文件数据，并将每个事件的能量与时间信息存储到文本文档中，用户在使用时根据具体情况修改对应内容。以下为关键步骤的程序语句：

1) 实现变量与数据的对应：

```
t1->SetBranchAddress("E", &E1);    // t1 是 Tree 的指针。
t1->SetBranchAddress("DeltaT", &DT1);
```

2) 创建文本文档 input.dat，用以存储导出的数据：

```
ofstream outfile("input.dat");
```

3) 获取 root 文件的总事件数，并逐事件运行：

```
Long64_t ntot1 = t1->GetEntriesFast();        //获取总事件数
for(i = 0; i < ntot1; i++)
      {
          t1->GetEntry(i);
         …
         outfile << DT1 << "      " << E1<<endl;
      }
```

2.2 拆分数据文档

本部分完整代码见附录 2。程序运行时需要读取 intervals.txt 文件，其中存储了时间间隔的数值，每两个数值为一组，依次为开始和结束时间，单位为 ns。拆分好的子文件依次以 001.dat、002.dat…进行命名，文件数量由时间的组数决定。为确保遵循数据处理原理，开始时间必须为 410 ns 的整数倍，并且每组的时间间隔均为 410 ns。

2.3 重组数据文档

本部分完整代码见附录 3。程序运行时需要读取 datfile.txt 文件，其中存储了所有子文件的文件名，包含后缀。重组后的数据文件依次以 add1.dat、add2.dat…进行命名，文件数量由数据组的数量决定。程序中设定的数据组上限为 45，如果数据组的数量超过 45，需要修改程序 34 行中 group 的上限。

2.4 获取能量单谱

在 2.3 节新生成的数据包括了时间及能量二维信息，其中第一列为时间，第二列为能量。获取能量单谱的方式为读取整个文件中的所有能量数值，并以用户规定的道宽统计每个道中能量数值出现的次数。完整的程序见附录 4，需要提取能量单谱的文件名存储于 filename.txt 文件中。能量的道宽在程序 30 行中修改。程序运行完成后会生成与源文件同名，但后缀名修改为.txt 的新文件。新文件中有两列数据，第一列为能量，第二列为对应的统计。至此，用户可获得单个束团在多个时间窗口内的能量单谱。

2.5 开发环境

操作系统：Windows 10 专业版

Root 版本：root_v6.30/04

C++代码编译器：Visual Studio 2021

3　结果与讨论

中子的入射能与飞行时间的平发呈反比，在时间道宽恒定的情况下，随着中子入射能的增大，中子能量的相对误差也呈平方关系增大。在 ES#2 终端测量中子俘获反应截面时，对于 10 keV 以上的能量区间采用 41 ns 为时间道宽，可以将中子能量的相对误差控制在 15%以内。本方法主要针对 10 keV 以下的能量区间，采用 4100 ns 为时间道宽也可以实现相同能量误差的控制。图 2 分别展示了以 4100 ns 和 41 ns 为道宽时的中子能量误差曲线。其中实线表示 4100 ns 道宽，点虚线表示 41 ns 道宽。可以看出，两种道宽在各自对应的能区内具有近乎相同的相对误差曲线。

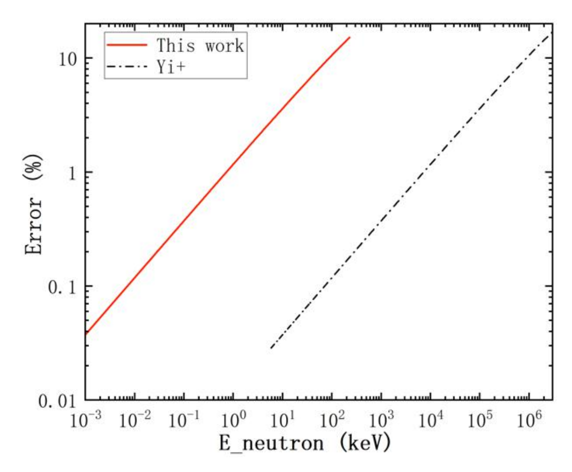


图 2　中子能量相对误差曲线。实线为 4100 ns 道宽的结果，点虚线表示 41 ns 道宽的结果。

Fig.2　Neutron energy relative error curve.

为了更好地展示中子能量误差的影响，本团队在 back_n 束流线上对 ^{91}Zr 的中子俘获截面进行了测量，时间范围从 11480 ns 至 191880 ns，时间道宽为 4100 ns，共 44 个数据点，对应中子的能量范围在 0.82 ~ 229 keV。利用本方法获得的中子截面如图 3 所示，从图中的横轴误差棒长度可以看出，当中子入射能超过 10 keV 时，中子能量的误差十分明显，相对误差最高可达 90%左右，中子俘获截面的变化趋势也相对平缓，难以判断是截面本身趋于稳定还是因为截面被平均后变得平缓，核反应截面的结构信息被掩盖了。当入射能逐渐减小时，中子的能量误差已控制到很小的数值，测量到的中子俘获截面震荡变得十分剧烈，很好地反映了该能量区间内的俘获截面共振特性。

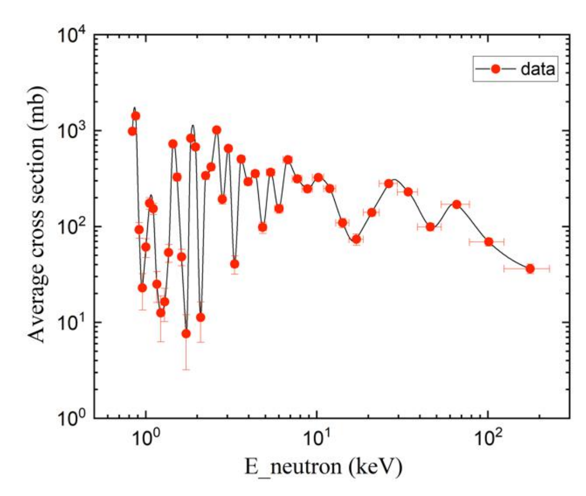


图 3　利用本方法得到的 ^{91}Zr 中子俘获截面。带 X 和 Y 误差棒的红点为截面数据。曲线为简单的连线，方便描述截面的变化趋势。

Fig.3　The Average cross sections of the neutron capture reaction of ^{91}Zr obtained using the proposed method in this work.

由此可知，当中子入射能超过 10 keV 且统计足够大时，使用精细的解谱程序可以获得更加精确的俘获截面，能准确体现反应截面的结构信息，而较宽的时间道宽会丢失这些结构信息。当中子入射能低于 10 keV 时，使用本方法同样也可以获得十分精确的激发函数曲线。

值得注意的是，本方法可支持的最小时间道宽为 820 ns，在保证中子能量分辨在 15%以内，最高可适用于中子入射能为 1.2 MeV 的数据。

4 结语

中国散裂中子源 back_n 束流线为中子俘获反应的截面测量提供了一个绝佳的实验平台。但是在常规模式下中子束流是由两个间隔为 410 ns 的束团混合而成的。为了得到准确的中子入射能，需要对双束团数据进行解谱重构。虽然已经存在一套精细的解谱方法，但是使用相对复杂，用户使用具有一定的门槛。而且对于需要使用较大时间道宽的数据，精细解谱方法的必要性不足。本工作根据基础的数学运算，提出了一种十分简化的解谱方法，以较大的时间道宽对数据进行拆分再重构。所有程序均可直接在 Windows 系统下编译运行，简单易用。

当使用最小时间道宽 820 ns 时，此方法在中子入射能低于 1.2 MeV 能区内得到的中子能量分辨可达到精细解谱法的水平，并且计算量得到了极大的简化。另一方面，相对精细解谱方法，增大了时间道宽，也可以提高每道的数据统计量，减小统计误差，对于统计不足的实验数据具有显著的优势。一言蔽之，对统计量足够大且中子入射能较高的数据，采用现有的解析方法更具有优势。而对于统计量较小以及中子入射能低于中低能区的数据，采用本方法具有计算量小，操作简单等优势。

参考文献

1 张杰. 中国散裂中子源(CSNS)——多学科应用的大科学平台[J]. 中国科学院院刊, 2006. **21**(5): 3. DOI: 10.3969/j.issn.1000-3045.2006.05.017.
ZHANG Jie. China Spallation Neutron Source (CSNS) - A Large Science Platform for Multidisciplinary Applications [J]. Bulletin of the Chinese Academy of Sciences, 2006. **21**(5): 3. DOI: 10.3969/j.issn.1000-3045.2006.05.017.

2 韦杰. 中国散裂中子源简介[J]. 现代物理知识, 2007, **19**(6) :8. .DOI: CNKI:SUN:XDWZ.0.2007-06-006.
WEI Jie. Introduction to the China Spallation Neutron Source (CSNS) [J]. Modern Physics Knowledge, 2007, **19**(6): 8. DOI: CNKI:SUN:XDWZ.0.2007-06-006.

3 Jing H T, Tang J Y and Tang H Q, *et al*. Studies of back-streaming white neutrons at CSNS[J]. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2010. **621**(1-3): 91-96. DOI: 10.1016/j.nima.2010.06.097.

4 陈延伟. 中国散裂中子源(CSNS)[J]. 中国科学院院刊, 2011. **66**(6): 726-728. DOI: 10.3969/j.issn.0368-6396.2014.04.005.
CHEN Yanwei. China Spallation Neutron Source (CSNS) [J]. Bulletin of the Chinese Academy of Sciences, 2011. **66**(6): 726-728. DOI: 10.3969/j.issn.0368-6396.2014.04.005.

5 Zhang L Y, Jing H T, Tang J Y, *et al*. Design of back-streaming white neutron beam line at CSNS[J]. Applied radiation and isotopes: including data, instrumentation and methods for use in agriculture, industry and medicine, 2018. **132**: 212-221. DOI: 10.1016/j.apradiso.2017.11.013.

6 Tang J, Liu R, Zhang G, *et al*. Initial years' neutron-induced cross-section measurements at the CSNS Back-n white neutron source. Chinese Physics C, 2021. **45**(6): 062001. DOI: 10.1088/1674-1137/abf138.

7 Hu X R, Fan G T, Jiang W, *et al*. Measurements of the ^{197}Au(n, γ) cross section up to 100 keV at the CSNS Back-n facility[J]. 2021. **32**(9): 101. DOI: 10.1007/s41365-021-00931-w.

8 Li X, An Z, Jiang, W, *et al*. CSNS Back-n Collaboration. Measurement of the ^{nat}Eu (n, γ) cross section up to 500 keV at the CSNS Back-n facility, and the stellar $^{151, 153}$Eu (n, γ) cross section at s-process temperatures. European Physical Journal A, 2022. **58**(12): 251. DOI：10.1140/epja/s10050-022-00887-4.

9 Chen Y, Qiu Y, Li Q, *et al*. Measurement of the neutron flux of CSNS Back-n ES# 1 under small collimators from 0.5 eV to 300 MeV. The European Physical Journal A, 2024. **60**(3), 63. DOI: 10.1140/epja/s10050-024-01272-z.

10 Yi H, Wang T F, Li Y, *et al*. Double-bunch unfolding methods for the Back-n white neutron source at CSNS[J]. Journal of Instrumentation, 2020. **15**(03): P03026. DOI: 10.1088/1748-0221/15/03/P03026

附录 1

```
#include <fstream>
#include <string>
void ana_extract(const char *fin1)//注意：函数名必须与.C 的文件名相同，否则报错
{
    Double_t E1;//能量变量
    Long64_t i; //事件数变量
    ULong64_t DT1; //飞行时间，单位为 ns
    TFile* fRun1 = new TFile(fin1);
    TTree* t1 = (TTree*)fRun1->Get("tr");   //创建 Tree 指针变量 t1，获取 root 文件中的 TTree
    t1->SetBranchAddress("E", &E1);
    t1->SetBranchAddress("DeltaT", &DT1);
    Long64_t ntot1 = t1->GetEntriesFast();   //获取总事件数
    ofstream outfile("input.dat");   //创建数据存储文档
    outfile << "DT" << "                    " << "E" <<endl;//在文档的第一行输出 DT 和 E
     for(i = 0; i < ntot1; i++) //逐事件处理数据
    {
        t1->GetEntry(i);
        if(i%(ntot1/10)==0)     std::cout<<"processing:"<<i*10/(ntot1/10)<<"%"<<std::endl; //显示数据处理进度，可删除
          outfile << DT1 << "      " << E1 <<endl; //将时间和能量数据写入文档
     }
     fRun1->Close();//关闭 root 文件，释放内存
     outfile0.close();
 }
```

附录 2

```
#include <iostream>
#include <fstream>
```

```
#include <sstream>
#include <vector>
#include <string>
#include <iomanip>
//创建数据的结构变量
struct Data {
    double DT;
    double E;
};
int main() {
    std::ifstream intervalsFile("intervals.txt"); //打开存储时间间隔的文件
    if (!intervalsFile.is_open()) {
        std::cerr << "Failed to open intervals.txt" << std::endl;
        return 1;
    }
    std::vector<std::pair<double, double> > intervals;
    double start, end;
   //读取时间的起点和终点
    while (intervalsFile >> start >> end) {
        intervals.push_back(std::make_pair(start,end));
    }
    intervalsFile.close();
    std::ifstream dataFile("input.dat");
    if (!dataFile.is_open()) {
        std::cerr << "Failed to open data.dat" << std::endl;
        return 1;
    }
    std::string title;
    std::getline(dataFile, title);
    std::vector<Data> dataVec;
    double dt, e;
    while (dataFile >> dt >> e) {
        Data data;
        data.DT = dt;
        data.E = e;

        dataVec.push_back(data);
    }
    dataFile.close();

    int fileCount = 0;
    for (const auto& interval : intervals) {
        std::vector<Data> filteredData;
        for (const auto& data : dataVec) {
```

```
            if (data.DT >= interval.first && data.DT <= interval.second) {
                filteredData.push_back(data);
            }
        }
        std::stringstream filename; //创建存储每个时间区间内数据的子文档
        filename << std::setfill('0') << std::setw(3) << ++fileCount << ".dat";
        std::ofstream outFile(filename.str());
        if (!outFile.is_open()) {
            std::cerr << "Failed to create output file: " << filename.str() << std::endl;
            return 1;
        }
        for (const auto& data : filteredData) {
            outFile << data.DT << " " << (int)(data.E + 0.5) << std::endl; //四舍五入将能量转换为整数
        }
        outFile.close();
    }
    std::cout << "Data filtering completed successfully." << std::endl;
    return 0;
}
```

附录 3

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
// 读取文件名列表
std::vector<std::string> ReadFileNames()
{
    std::ifstream inputFile("datfile.txt");
    std::vector<std::string> filenames;
    std::string filename;
    while (std::getline(inputFile, filename))
    {
        filenames.push_back(filename);
    }
    return filenames;
}
// 将源文件的内容追加到目标文件
void AppendFileContent(const std::string& srcFile, const std::string& destFile)
{
    std::ifstream src(srcFile, std::ios::binary);
    std::ofstream dest(destFile, std::ios::binary | std::ios::app);
    if (!src || !dest)
```

```
    {
        std::cerr << "Error opening files!" << std::endl;
        return;
    }
    dest << src.rdbuf();
}

int main()
{
    std::vector<std::string> filenames = ReadFileNames();
    for (int group = 0; group < 45; ++group) //根据数据组的数量修改 group 的上限
    {
        std::string destFilename = "add";
        destFilename.append(std::to_string(group + 1));
        for (int i = 3 - destFilename.length(); i > 0; --i)
        {
            destFilename.insert(destFilename.begin(), '0');
        }
        destFilename += ".dat"; //添加文件后缀名
        std::ofstream dest(destFilename, std::ios::binary);
        for (int i = group * 10; i < std::min((group + 1) * 10, static_cast<int>(filenames.size())); ++i)
        {
            if (i % 2 != 0) //  间隔取子文件
            {
                AppendFileContent(filenames[i], destFilename);
            }
        }
    }
    return 0;
}
```

附录 4

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <map>
#include <string>
#include <vector>
int main() {
    //  输入文件流
    std::ifstream filenameFile("filename.txt");
    if (!filenameFile.is_open()) {
        std::cerr << "Failed to open filename.txt" << std::endl;
```

```
            return 1;
        }
        //  输出文件流
        std::vector<std::string> filenames;
        std::string filename;
        while (std::getline(filenameFile, filename)) {
            filenames.push_back(filename);
        }
        filenameFile.close();
        //  统计每个间隔的计数
        for (const auto& filename : filenames) {
            std::ifstream file(filename);
            if (!file.is_open()) {
                std::cerr << "Failed to open file: " << filename << std::endl;
                continue;
            }
            std::map<int, int> countMap;
            int dt, e;
            int bin = 20; //能量道宽
            while (file >> dt >> e) {
                int interval = (e / bin) * bin;    //取 bin 后
                countMap[interval]++;
            }
            file.close();
    //创建新文件
            std::string outputFilename = filename.substr(0, filename.find_last_of(".")) + ".txt";
            std::ofstream outputFile(outputFilename);
            if (!outputFile.is_open()) {
                std::cerr << "Failed to create output file: " << outputFilename << std::endl;
                continue;
            }
            for (const auto& pair : countMap) {
                outputFile << pair.first << " " << pair.second << std::endl;
            }
            outputFile.close();
        }
        std::cout << "Data counting completed successfully." << std::endl;
        return 0;
    }
```